Testbed Infrastructure for Debugging, Analyzing and Optimizing WSN Nodes Based on a Modular HW-SW Architecture

Gabriel Mujica, Jorge Portilla, Teresa Riesgo
Centro de Electronica Industrial, Universidad Politecnica de Madrid
Email: {gabriel.mujica, jorge.portilla, teresa.riesgo}@upm.es

***Abstract*—The Internet of Things has emerged as one of the key aspects to the future of the Wireless Sensor Networks and their impact in new applications in real environments. This concept poses new challenges in the implementation, testing and assessment of efficient, robust and reliable technologies and prototypes under this paradigm. In this way, the run-time remote interaction with the deployment of hundreds of in-field nodes in which developers have to be able to control and manage the wireless network anywhere at any time also implies new objectives to be achieved in order to adapt or even create new HW-SW platforms. In this work, the design and implementation of a complete testbed infrastructure as a support tool for improving the effectiveness and the applicability of sensor nodes to real applications is presented, focused on the modular architecture of the Cookie hardware platform and aiming to help developers to integrate and optimize the whole WSN system to the final applications in the real world.**

Keywords — wireless sensor networks, testbed infrastructure, HW-SW platform, backchannel architecture, node modularity.

I. INTRODUCTION

During the last years the paradigm of the Wireless Sensor Networks (WSN) and the Internet of Things (IoT) has been strongly involved in most of the new technologies and research lines for different application fields, such as smart cities, industrial environments, security issues, energy harvesting and home automation, among others, aiming to interact to the environment remotely anywhere at any time. One of the key topics in which the concept of IoT is having more and more impact is in the WSN applications [1], where hundreds or even thousands of small devices have to be accessed, managed, controlled and monitored in an efficient and reliable way, so that feedbacks from in-the-field deployment are to be obtained. However, the challenges that involve the Wireless Sensor Network technology, such as power consumption and long term autonomy, low-data-range based wireless protocols for sensor nodes, limited bandwidth, limited processing resource and limited memory, pose much more challenges and new goals to be covered in order to effectively expand and validate the applicability of this challenging topic in real scenarios.

In this way, new research lines have been recently growing in order to experiment with new implementations of technologies, focused on optimizing the Wireless Sensor devices and increasing the efficiency of algorithms and protocols that are to be included on the hardware platform [2]. Therefore, the creation of novel testbed infrastructures must be enclosed by flexible architectures and the ease-of-use on the management, monitoring and reconfiguration of sensor nodes remotely from any place, in order to run experiments for testing new approaches and developments. Most of the current testbed platforms are based on commercial hardware which are widely used but start being obsoletes and much less flexible and reliable for new technologies that come out with this new world of the WSN and the IoT. Besides, the old node platforms that are used in most of the available testbeds are limited in case of testing new hardware technologies, due to the lack of well-defined hardware debugging capabilities. These issues motivate the creation of a complete new testbed infrastructure in order to run experiments related to this novel approach within the WSN technology, by continuing developing the Cookie Nodes technology [3] which offers a complete HW-SW integration platform that accomplishes key features to the future of the WSN within the IoT: Modularity, flexibility, scalability, heterogeneity and repeatability.

The work proposed in this paper is then focused on having a complete feedback to a real Cookie-based WSN by means of combining a high performance backchannel infrastructure (Based on Ethernet and Wi-Fi connectivity) with the design of a novel memory-segmentation-based architecture to allow users to partially and remotely reprogram their experiments in a dynamic way, by providing a mechanism for modifying specific components of the system, more than just reprogramming the whole running application. Moreover, new software libraries for increasing the robustness of the HW-SW platform are also proposed, in order to help users to debug and experiment in runtime with a real deployment scenario in an easy but reliable way. This laboratory infrastructure is part of CookieLab: A Wireless Sensor Network Testbed that is being developed at the Center of Industrial Electronics (CEI - UPM). As a pre-deployment tool, the testbed helps users to optimize prototypes from the laboratory stage before implementing them in the final application, in order to assure the efficiency and effectiveness of the nodes.

The rest of the paper is organized as follows, starting with an overview of the related work in section 2. In section 3 a deeper discussion of the proposed architecture is presented, whereas in section 4 more details regarding the design and implementation of the testbed infrastructure is approached in details. In section 5 experimental results are analyzed and finally conclusions and future works are presented.

II. RELATED WORK

Several testbed platforms have been proposed during the last years, focusing on testing prototypes and applications before final implementations. One of the most well-known testbeds is Motelab [4] that is an open access platform for testing experiments based on MicaZ motes. Nodes can be accessed by a PHP web server. In Kansei [5] the concept of simulation over a testbed support platform is proposed, by implementing 210 sensor nodes, which are connected to Linux-based stargates that control the data collections. SignetLab [6] implements 48 EyesIFXv2 motes that are

connected to tiered USB hubs which are also connected to a single PC in order to control, manage and reprogram the deployed nodes by using a Java based application. In terms of testbeds focused on final application scenarios, different approaches have been presented in recent years, such as WINTeR [7] in which a full scale industrial deployment is proposed in order to evaluate and support industrial applications, based on programmable motes controlled by a web interface. This testbed is based on two backchannels, the first one for controlling the data exchange to the nodes (Ethernet connection) and the second one for powering the nodes (USB connection). In [8] an in-field tracking and localization testbed was developed in a manufacturing lab scenario in order to model the effects that can be found in real deployments with similar conditions, by using TelosB connected to a central server which runs a java-based software application to control the experiments. There are also different approaches that propose the management and control of the testbeds by using the main wireless link, without then implementing backchannel infrastructures, such as the work presented in [9], where a software component to be installed in the nodes is proposed combined with a web interface in order to control and manage the deployment using the wireless link.

Although the use of the main wireless link as a debugging channel reduces hardware costs, this approach is bandwidth-consuming and does not allow performing a real evaluation of node/network features such as energy consumption, or communication stack and protocol analysis, since all the debugging actions are totally intrusive due to the WSN protocol resource usage for this purpose. Moreover, most of the current testbed platforms aim to test software prototypes by using the well-known TelosB or MicaZ nodes, which are becoming obsolete due to its low flexibility and their limited adaptation to hardware expansion and testing. The main way of performing evaluation tasks in these types of platforms is by using *printf-based* messages over a serial interface from the node's microcontroller. Apart from it, the capabilities of dynamic experimentation by performing remote and partial reprogramming of the WSN testbed platform is not covered in most of the available systems, or is only done by means of TinyOS component replacement, which is totally oriented to developments under this operating system.

III. System Architecture – Considerations, Requirements and Challenges

Based on the proposed concept of a WSN Testbed platform which is aimed to provide developers and researchers with a complete set of HW-SW support tools beyond simulation capabilities, this testbed infrastructure is focused on allowing users to test their prototypes, new algorithms and built-protocols in a real environment (more than just verify that a sensor works as expected), based on a completed pre-deployment scenario in where the platform can be programmed and configured with parameters related to the experiment to be performed in runtime. The results of the tests are based on real measurements and the real behavior of the system, not as an interpolation of modeled behaviors, which is the case of WSN simulations, where assumptions are done due to computational and modeling limitations. Users can then access to the parameters of every node, gather actual data of the hardware platform, assess the network performance and, in case of needed, modify configurations in order to change measurement parameters or verify functional blocks. By using the proposed backchannel-based architecture, developers are provided with a way to interact to the sensor nodes and the whole network without interfering to the main wireless link and, therefore, without mixing the information related to the application with the debugging and analysis tasks. A non-intrusive mechanism is therefore taken into account as a design target, in contraposition with the limitations of the main-network-based/backchannel-free testbeds.

In order to face such a challenging goal, four main aspects and considerations have to be covered for the success of the testbed architecture that is being proposed, as split below.

Remote access & reprogramming – One of the key aspects related to the testbed capabilities is the ability to modify not only parameters regarding configurations of the nodes, but also entire algorithms, functional blocks or even the whole application in an remote, efficient and reliable fashion. In this context, the use of an independent channel to replace or download programming files into de core of the nodes without using the main wireless link takes place. The backchannel shall allow modifying the memory flash of the devices where the experiments are running, with no interference with the wireless network itself. Mechanisms regarding how to access remotely the memory and the HW-SW interface to download specific components make this reprogramming procedure more challenging.

Energy Assessment – one of the research targets in WSNs is the energy efficiency, not only regarding hardware platforms' power consumption, but also how algorithms and protocols can be optimized in order to enhance as much as possible the autonomy of the node, and therefore extend the lifetime of the systems in their final operation. In this way, the testbed infrastructure has to provide the ability to analyze the performance of the nodes in terms of energy consumption in runtime, in order to define the energetic behavior of specific prototypes as well as apply strategies to switch on/off nodes so that the self-reconfiguration of the network in terms of topology changes can be monitored remotely (which is also intended to evaluate implementations such as routing protocols and MAC algorithms). In order to accomplish this, the platform has to be capable of providing not only strategies to power supply the nodes, but also measurements of the power consumption of the devices remotely whenever it is necessary.

Debugging capabilities – As previously highlighted, the idea of a WSN testbed platform is not just verify and validate that a sensor is gathering data within a predefined threshold. In order to evaluate and verify the robustness and effectiveness of a WSN, it is necessary to include mechanisms for system debugging, from the hardware layer up to the application level, including the behavior of internal peripherals, interfaces among HW components and software functionalities. Moreover, the interaction among the nodes is also a key aspect to be considered, which introduces the concept of network debugging. A local correctness of node functionalities could not then imply the validation of the system as a whole. Therefore, in this work three levels of system debugging are proposed: hardware level, software-components level, and network-application level.

Multi-experiment – Partial reprogramming strategies – since testing, prototyping and assessing new WSN technologies, algorithms and protocols is an iterative process where optimization are performed based on the results of the debugging tasks, the remote reprogramming needs to be performed in an efficient way so that changes in the

experiments shall not imply a long and resource consuming process every time it is carried out. It could cause unnecessary failures during transmission and prolong the dissemination stage of the components under tests. On the other hand, if the memory shall be written every time an experiment is set to be tested rather than downloading several test units during the same dissemination stage, the performance of the testbed decreases in terms of flexibility, operability and usability of the platform. Therefore, in this work a novel mechanism to create a multi-experiment architecture based on the concept of partial and remote reprogramming of the nodes is included, in which specific components of the system can be modified, and several tests units can be downloaded and scheduled under the same node, so that an experiment repository can be reprogrammed and time slotted within the same device.

Based on these considerations that also define the scope of this work, a novel architecture for testing, debugging and assessing WSN technologies is proposed, focusing on optimizing the performance and efficiency of the Cookie platform with the support of a set of HW-SW developments and components. A general overview of the proposed platform is shown in figure 1. In order to accomplish the design of the WSN testbed infrastructure in terms of hardware and software components, four main developments integrate the overall platform fitting the aforementioned requirements, as described in the following sub-sections.

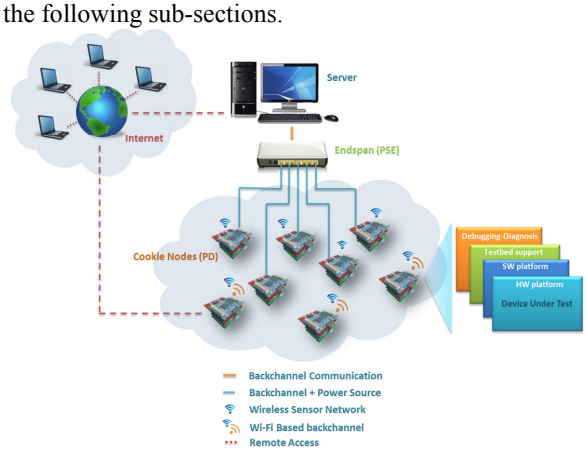

Figure 1. General view of the proposed Architecture.

3.1. Backchannel interface and testbed communication

The testbed infrastructure, as a development and testing tool, shall be capable of providing reliable communications from the devices under tests to a remote access, so that the robustness of the backchannel is completely assured for the performance of the experimental tasks. In this way, two main high performance based communication layers are included to satisfy the remote access requirements.

On one hand, a combination of the standard-based Ethernet connection with the concept of Power over Ethernet (PoE) is included by means of designing an integrated Cookie-compliant layer. So let's first analyze three main concepts that have to be considered in order to use this technology and its implications in the testbed architecture. First of all, the idea of using Ethernet is not only related to the reliability of the communication and the bandwidth (apart from its ease of integration in an indoor-based testbed by taking advantage of the already deployed intranets in lab environments), but also to include the *PoE* technique, which takes advantage of the physical connection of the Ethernet standard to transfer power to end devices by using two pair wires that are not part of the data signals, so that power supply and data can be obtained from the same cable, reducing the complexity of the interconnections. The Ethernet-based connection brings the possibility of controlling the sensor nodes remotely either from any point of the intranet or even from outside of the laboratory by accessing the testbed platform through internet.

Secondly, the approach of Power Source Equipments (PSE) shall be taken into account. These are the devices in charge of acting as both the routers of the Ethernet communication and power supply providers through each available socket. In this way, it is important to highlight that the PSE can be either a device that receives the Ethernet connections from other switch and adds the powered signals to the incoming data signals (in this case the device is called Midspan), or a device that receives the Ethernet input connection and switches the signals to its available output with the power supply wires prepared with the PoE standard (this device is called Endspan). Aiming to have as less complexity of the interconnection as possible, the selected configuration to the PSE has been the use of an Endspan device. Apart from the power supply capabilities of these elements, the switching actions on the device to be powered are done at this level, so the PSE provides with the possibility of turning on/off the nodes remotely.

Finally, the Powered Device (PD) is considered as the final nodes that receive both data signals and powered signals from the PSE through the same cable. As defined in the PoE standard, it is possible to obtain up to 54 V and up to 15.4 W from the Ethernet port which completely covers the power supply requirements of the end devices (sensors, actuators, routers, etc). In case of the proposed architecture, the powered devices are the Cookie-Nodes, which are the final devices under test. Within the implementation of a Cookie-compliant PoE device, the conversion of the high input voltage levels of the Ethernet port to the standard levels of the Cookie layers is done (3.3V/2.8V, 2.5V, 1.8V 1.5V and 1.2V, for components such as the FPGAs, microcontrollers, among other internal elements included within the system), as well as the inclusion of the measurement of the power consumed by the platform, so that the energy behavior can be analyzed in runtime. Moreover, the Ethernet stack is also included so that the remote actions are transparently performed over the backchannel by using a standard interface, such as SPI or UART, as shown in fig. 2.

On the other hand, to tackle the limitations of the Ethernet in terms of mobility of the sensor nodes as well as the scalability and flexibility of the platform, apart from the performance evaluation in outdoor environments, the inclusion of the IEEE 802.11 within the testbed infrastructure is also proposed, which enhance one of the key features to be covered

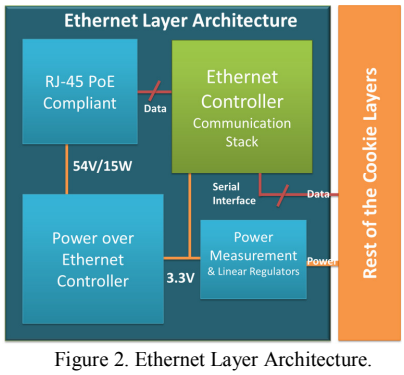

Figure 2. Ethernet Layer Architecture.

in these scenarios: interoperability and heterogeneity.

A high performance wireless protocol for performing the backchannel debugging tasks also implies several advantages in the support of final in-field outdoor scenarios. First, in those scenarios where Wi-Fi connection is limited in certain areas, cluster nodes can be defined as heterogeneous devices that implements the low-rate based communication protocol for the wireless sensor network, and the high performance connection to interface the remote server. On the other hand, in both indoor-outdoor-based testbed scenario, the physical topology in terms of localization and placement can be easily modified, which provides developers with the ability of testing mobility, routing performance, coverage and planning algorithm problems in sensor networks.

Similar to the Ethernet-based Cookie layer, a new communication layer for high level wireless connectivity is then proposed, which includes the integration of a Low-Power Wi-Fi module that implements the stack to be accessed from a high abstraction level by using a standard interface, such as UART. This configuration is also focused on the ease of integration of different communication protocols within the same architecture, taking advantage of the modularity of the Cookie platform. Hence, the design of a heterogeneous WSN testbed infrastructure is proposed in order to maximize the capabilities of the remote experimentation schema, where four main communication technologies converge: Ethernet, IEEE 802.11, IEEE 802.15.4 and ZigBee, merging robustness and flexibility with modularity in a unique support platform.

3.2. Hardware support platform – debugging blocks

Unlike works where the main way of debugging remotely the sensor nodes is by performing *printf-based* messages from the node's microcontroller, the present work addresses not only a heterogeneous backchannel interface for debugging tasks, but also the capability of carrying out this actions by accessing remotely specific hardware and software elements of the modular platform, i.e., capturing specific signals of interest such as GPIOs of the node core, sensor interfaces and their status, and communication interfaces among the different layers. Moreover, the ability to analyze specific software components by including a remote debugger for the microcontroller of the platform helps developers to cope with specific failures or application malfunctions that cannot be detected by simple *printf* usage. In this way, bringing the possibility of carrying out run-time debugging tasks by capturing the status of specific registers, memory sectors, flag-masks parameters and peripheral configurations, developers are able to modify the behavior of their experiments, more than just gathering data related to the application running. Thus, the main target of this proposed debugging system is the possibility to test sensor nodes from different abstraction levels, from the hardware-signal-interface level up to the application level.

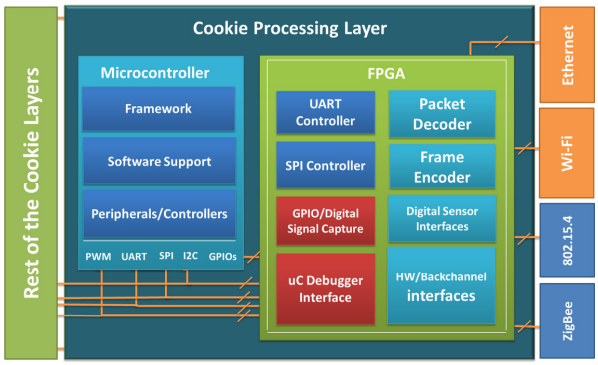

Figure 3. HW-SW debugging schema.

In fig. 3, the proposed debugging scheme is shown in detail, where the FPGA is connected to the majority of digital signals and interfaces involved in the Cookie platform. By combining the hardware blocks implemented in the FPGA with the debugging connection to the microcontroller and, therefore to its internal peripherals, the testbed can have control of both hardware and software elements.

On one hand, the debugging blocks can obtain information related to the digital interfaces of the sensors, such as I2C, 1-wire, PWM, as well as the status of the generic interfaces proposed in [10]. On the other hand, the interfaces related to the connection from the microcontroller to the communication modules can also be gathered and filtered by the FPGA blocks in a transparent way, i.e., without interfering in the behavior of those lines, so that specific signal levels shall be analyzed in order to find low level problems during information exchanging. Moreover, the advantage of having such a hardware element provides the ability to switch on/off specific signals in order to inject test signal levels or event failures, so that the response of particular functional blocks to controlled actions can be experimented.

This is the case of testing, for instance, protocol implementations regarding communication layers, such as an AODV-based routing protocol on top of the IEEE 802.15.4 MAC and PHY layers. In this particular case the testbed debugging block injects locally a specific packet including the corresponding MAC frame in order to stimulate a specific component of the implementation to be evaluated. Since every Testing Action has its corresponding code number (called Test Identifier, which includes the input information of the testing process received from the testbed server side, as a reference of the action to be performed), the debugging block is in charge of generating the subsequent Action Frames in order to provide developers with the result information according to the response/behavior of the component/functionality under test. The actions to be performed can be classified depending on the type of debugging level, as follows:

- GPIO/Signal Level
- Sensor/Actuator Level
- Interface/Communication Level
- Component/Peripheral Level
- Implementation Level
- Application Level

Every action/test will automatically generate an Action Frame by the debugging block in the corresponding encoded format, according to the input information of the Test Identifier. The Action Frames are then sent through the proposed backchannel interface.

Regarding the debugging capabilities of the specific internal architecture or software functionalities related to the microcontroller, it has been necessary to integrate a custom debugging component in order to remotely perform actions in runtime. In this way, since most of the available development tools support debugging capabilities by using standard interfaces, the remote debugging of the microcontroller is done by means of JTAG, UART or SPI, and the corresponding conversion to the backchannel interface is then carried out, which process is transparent from the user point of view. Thus, by including a small code segment in the uC for the

debugging tasks, the capabilities of available IDEs that support the corresponding core (in this case the well-known and extended 8051-based architecture) can be used.

3.3. Memory-segmentation-based architecture

As previously explained, the concept of splitting the available program memory into modular segments to remotely reprogram it in a partial fashion is introduced, so that different and independent testbed experiments can be downloaded and scheduled to be run at different times. More than just organizing the memory size and the code location, a novel segmentation-based architecture is proposed in order to support such a challenging operational schema.

Focused on the 64KB-based flash memory of the 8051 architecture, four main operational areas have been defined, as classified below:

Protected–Recovery area: this segment includes a custom bootloader to be able to receive the reprogramming file remotely and write it in the corresponding memory slot defined for it. Moreover, a recovery mechanism is also considered so that in case one of the segments of the memory map fails, a predefined system application is launched in order to allow developers to execute a segment repair/replace process. This segment is allocated in a fixed position of the memory map and defined as write-protected, so that the integrity of the system is totally assured since it cannot be modified in a normal operation of the testbed platform.

Support Area: This segment is in charge of providing user's experiments with the support of the HW-SW architecture of the Cookie platform, which means that the framework containing the set of libraries and components related to the software support platform is included in this operational area. Although this segment is reserved for system supporting and internal debugging, it is possible to modify the organization of this area in order to include new libraries for the management and control of novel hardware developments. In this way, the area of interest is calculated as follows:

$$A_L = \sum_{n=O}^{Ns-1} (L_{Sn} + O_{Sn}) \quad (1)$$

A_L is the reserved area for the software support platform, which contains the libraries and components to be used. L_{Sn} is the size of the slot n, which depends on the total sub-area of every library in its current version. N_S is the amount of libraries to be included in this area, whereas the O_{Sn} is a generic offset which is assigned to every slot in order to include replacements, modifications or changes in the current version of the libraries. Applying the same offset in every library slot, O_s, the expression remains as follows (note that every offset can also be calculated by assigning a weight depending on the size of the current library):

$$A_L = (O_S * N_s) + \sum_{n=0}^{Ns-1} L_{Sn} \quad (2)$$

Two versions are available, the first one aimed to be used in final implementations where the size of the segment is optimized, not assigning O_s to any sub-area, so the first element of the right side of the expression is null, resulting on a smaller size of the support area. In this case, the target of the experiments is related to the usage of the dynamic area for testing algorithms, applications, etc. on top of the support area, which thus stays fixed. The second version is intended to be applied in platform experiments where modifications of the support libraries are carried out. In this scenario, every available sub-area for library updates is calculated as follows:

$$A_{Sn} = L_{Sn} + O_{Sn} + O_{Sn-1} + O_{Sn+1} \quad (3)$$

A_{sn} is the maximum size per slot that can be updated. Every time a slot is updated with a new version of a library, L_{sn} and O_{sn} have to be updated as well, so that every time a sub-area shall be updated, the adjacent slots will not be corrupted. With this expression the size of the support area is optimized to minimize the insolated memory space, without being necessary to reprogram the whole memory sector, maximizing the capabilities of the partial reprogramming mechanism.

Management Area: This segment is the core for establishing and controlling the correlation between the recovery and support segments with the application area where users download their corresponding experiments. Three main aspects can be differentiated in this context. First, a standard mechanism to link library calls/usage of user applications to the support platform has to be done, since the libraries will be already stored in the sensor node whenever users reprogram remotely their application experiments (It is not necessary to include the library files in their projects). Hence, in order for them to be transparent concerning memory position of the libraries, when the user application calls a component the system will automatically jump to a position of the management segment, where the actual position of the component is then executed. In other works, the management segment has the final position of the libraries whereas the user application includes a relative call. This mechanism aims to avoid changing the user application code in case the location of the support libraries changes, since the application sees the functions always in the same memory location, therefore avoiding reprogramming the whole segment every time a component/library is updated-modified. Secondly, since every application can handle their own runtime interruption functions, a similar mechanism to the aforementioned is used to link them to the real position of the interruption vector defined in the core architecture, as represented in fig. 4. On the other hand, from the point of view of the execution of different experiments downloaded into the testbed platform, an application scheduler is included, which controls the running process of the user's segment based on a preprogrammed scheduling. Users can assign time slots for different experiments and the management segment will automatically launch the corresponding application according to the stored information in the scheduler.

User Area: This is the dynamic segment to be used by users, where memory slots are assigned to allocate different experiments at the same time. Therefore, this is the memory sector that is usually reprogrammed remotely and partially. A similar approach as the one described for the support area is followed to calculate the memory slot reservation, but in this case the area per slot can be modified depending on the size of the experiments, which means that N_s changes much more than in the case of the support libraries.

A general overview of the proposed memory map architecture is shown in fig 4. It is important to highlight that this scheme is totally transparent for users when they develop their custom applications/experiments by using the Cookie support platform. All the directives and low level definitions are encapsulated in a provided framework in which they create their algorithms and tests by using the support platform.

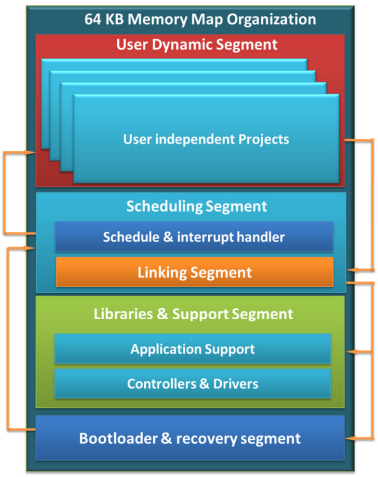

Figure 4. Memory-segmentation-based partial reprogramming architecture.

3.4. Software support platform – software components

As it was proposed in [11] a HW-SW integration platform for managing and controlling Cookie Nodes in real applications was designed, focused on providing a complete framework for users to implement new prototypes, algorithms and designs based on this technology. The aim of an ease-of-use support platform for addressing the key features of a Wireless network deployment has been carried out, based on the experience of real requirements that can be found within in-field scenarios. However, the implementation of the testbed infrastructure proposed in this work posed the need of extending the HW-SW platform in order to assure completely the integration of the backchannel communication interface as well as the management of the debugging, maintenance, and reconfiguration tasks that have to be supported. Following the concept of the Cookie HW-SW platform, the main target is to be able to interact to the system from a high abstraction level, providing not only low level controllers for the designed hardware, but also functions for managing connectivity to the nodes, preconfigured files for handling the standard protocols by setting up parameters into the modules, functional blocks for reprogramming the nodes with new configurations and experimental algorithms, among others features, hence helping developers to use properly the whole system in an easy but effective way.

In this way, a complete set of software libraries for controlling both the Ethernet and the Wi-Fi based backchannel has been implemented, so that users only have to configure few parameters in order to run a new experiment into the testbed architecture. More than this, the design and creation of functional blocks for allowing the runtime remote reconfiguration of each node through the laboratory communication were carried out and implemented. These modular components are included in addition to the framework that provides users with libraries for controlling every aspect of the hardware platform, such as communication modules (ZigBee, IEEE 802.15.4) sensor's repository, internal peripherals, etc. as well as the implementation of algorithms and protocols such as a modified AODV-based routing protocol which works on top of the 802.15.4 stack.

Since the main target of debugging WSN based applications is not only the assessment of every single node but also the network as a whole system, in this work the idea of integrating network and platform diagnosis by implementing software components in combination with the support platform is also introduced. This test component complements the evaluation of the hardware-software platform, by carrying out experimental actions in order to analyze the performance of the wireless network in terms of connectivity matrix, routing protocol robustness, node synchronization, medium access efficiency, coverage, among others. Combined with the capabilities of planning tools, network diagnosis helps users to optimize deployment strategies and key aspects of WSNs for the success of the final application. Three main diagnosis tests are proposed to analyze the quality of the network deployment, which are described as follows:

MAC layer assessment–Connectivity matrix: in order to analyze node coverage by obtaining the information regarding the quality of the radio signal and link with its neighbors. By carrying out this test, a complete connectivity matrix in terms of PHY and MAC layers can be built in order to evaluate not only the efficiency of the nodes related to their current deployment position, but also enter in low level details regarding routing protocol decisions, as shown in fig. 5. The coverage is analyzed by using two parameters: the Link Quality Indicator (LQI), and the Radio Signal Strength Indicator (RSSI) in both ways (from source to destination and also the other way around), which helps users to study in detail path symmetry in real scenarios as well as metrics efficiency in routing discovery.

Network Map: which obtains routing table information of every node in order to trace and analyze bottle necks by computing Packet Loss Rate (PLR), as well as evaluate discovery decisions in routing protocols. Five main parameters are considered to analyze paths: Next hop, number of hops of the built path, destination node, PLR of the current table entry, and route metric. By comparing the network map with the connectivity matrix, conclusions regarding the implementation of routing protocols can be extracted from the experiments, combined with key parameters regarding the performance of a sensor node such as the energy profile of the device.

Path Assessment: where a configured number of packets is disseminated through the network in order to analyze the ratio of Transmitted/Received packets from the source nodes to the destination points. In this scenario, two major features can be tested: first, network traffic, in order to detect congestions and routing robustness; and second, path variability, in which the creation of alternative routes to rich the destination nodes are detected, so that coverage/saturation problems can be studied.

The set of included libraries to support the described capabilities of the testbed infrastructure is presented in the software implementation subsection.

IV. IMPLEMENTATION

In order to address the aforementioned requirements and the proposed system architecture, both hardware and software implementations have been carried out to support the testbed platform, as described in the following sub-sections.

4.1. Hardware infrastructure

Focused on the definition of the Ethernet-based backchannel communication, two elements are to be

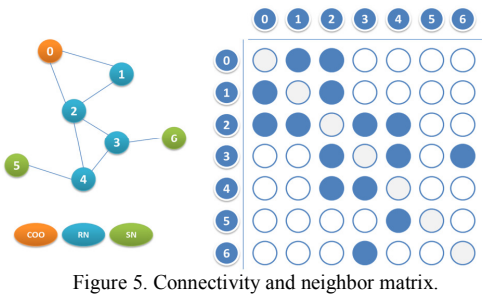

Figure 5. Connectivity and neighbor matrix.

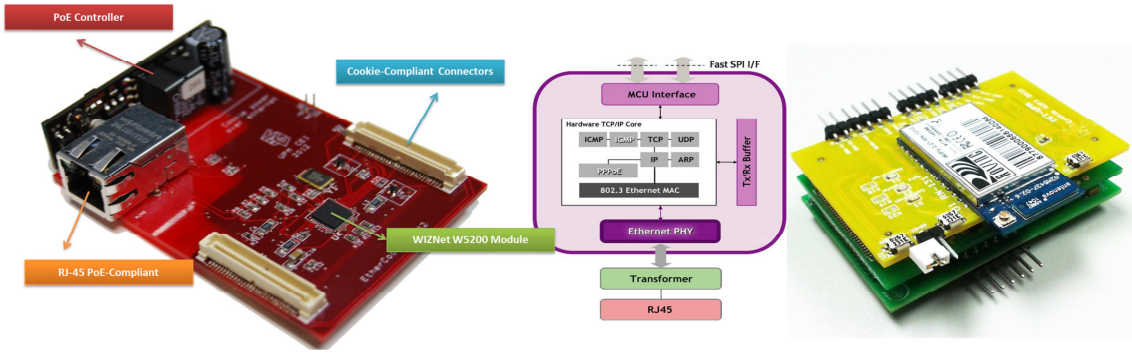

Figure 6. Implementation of the backchannel architecture.

considered in the implementation schema. First, commercial Endspans have been evaluated depending on the number of available ports per switch and also the power that the devices are able to supply per connection. This value is a key point in the design because of the power consumption of the final nodes. Therefore, the selected one has been the Netgear GS110TP switch [12], which supports the PoE standard as well as 10/100 Mbps (Ethernet and fast Ethernet, respectively). Secondly, the design of the Cookie-based Ethernet Powered Device has been carried out in this work. The design and implementation of this concept has been focused on covering three key aspects of the proposed architecture in a unique device. First of all, a Cookie-compliant design has to be taken into account, which aims to the flexibility, modularity and robustness of the architecture. Second, the platform must be compliant with the PoE standard as Powered Device and thus being able to convert the input levels of the power supply coming from the Ethernet port in order to provide energy to the modular layers that are part of the Cookie node. Third, an adequate interface between the processing layer of the Cookie platform and the data signals of the Ethernet standard has to be established, so that the node is able to exchange information to the server by using that interface (through standard protocols such as TCP/IP).

The final implementation of the proposed Cookie-Ethernet-platform is shown in fig. 6 (left), named EtherCookie**.** As mentioned before, the voltage supply level coming from the Ethernet port must be adapted to the operation levels of the Cookie layers. In this way, two stages in the power supply system design have been followed in order to address these requirements. On one hand, a PoE controller for adapting up to 54 V to 3.3 V has been included in the implementation. On the other hand, linear regulators for supplying the different required voltages have also been included in the proposed layer. Several commercial PoE controllers were evaluated to fit the requirements, and the selected module was the Silvertel Ag9403-2BR from the Ag9400 family [13], in which nominal output voltages can be found, such as 24V, 12V, 5V and 3.3V. For this particular case, the Ag9403 provides 3.3V and 6.6 watts as maximum output power that is enough for the requirements of the Cookie nodes. It is also important to highlight that not any type of RJ-45 is appropriate to be used as PoE socket, because most of the commercial connectors do not have access to the lines that have to be connected to the powered signals (instead, they are connected to ground). Therefore, it has been necessary to include a RJ-45 PoE-compliant, such as the BelFuse SI-52008-F [14] selected for the final implementation. In terms of data exchange through the backchannel, a SPI-Ethernet interface solution, which allows achieving up to 250 Kb/s, has been adopted by selecting the WIZNet W5200 module [15]. This module supports most of the standard protocols for Ethernet communication, such as TCP, UDP, IPv4 among others, having High Speed SPI, as shown in fig. 6 (middle), in which the low level communication layers (PHY layer, MAC layer, network and transport layer) are implemented, so that the configuration and management of the Ethernet communication is done by the processing layer from a higher abstraction level.

Regarding the integration of a Wi-Fi-based node as part of the testbed infrastructure based on the modularity of the Cookie platform, a new communication layer for high level wireless connectivity is proposed. In order to achieve this objective, commercial Wi-Fi modules have been analyzed regarding Cookie-compliant features (such as power source, interface, power consumption, etc.), in order to be included in the hardware platform. As a result of this comparative analysis, the RN-131C from Roving Networks was selected to be integrated in the final implementation. This module addresses those key features: IEEE 802.11 b/g as implemented protocol with different authentication modes such as WEP-64 and WPA2-PSK, typical output RF power = +18dBm, Power Source = 3.3 V, several power consumption modes, having 4 uA in sleep configuration, on-board chip Antenna with the possibility of integrating an external one, UART interface with AT command-based configuration, and several functionalities such as Timers, GPIOs, 8 Mbit flash memory and 128 kB RAM [16]. The design of the layer follows the standards and dimensions of the Cookie platform, as shown in fig. 6 (right) where the hardware modularity is highlighted.

4.2. Software Support Platform

Apart from the existing integration platform, nine new software component packets are proposed in this work, in order not only to support the testbed infrastructure, but also to test and experiment with the proposed architecture by integrating several test cases. Moreover, a new framework is also taken into account in order to provide users with the capabilities of the memory segmentation schema. A brief description of the included libraries is presented as follows.

CEI_Ethernet: this packet covers the functionalities regarding the configuration and maintenance of the backchannel form the point of view of the EtherCookie implementation. Functional blocks include W5200 configuration, protocol stack configuration, sockets configuration and management, buffers and queues control, and interruption configuration.

CEI_WiFi: similar to *CEI_Ethernet* for the support of the testbed backchannel infrastructure, including Access Point and Ad-hoc network configurations, several types of authentications (such as WPA2-PSK or WEP-128), sleeps modes and TCP/UDP connectivity, among other functionalities.

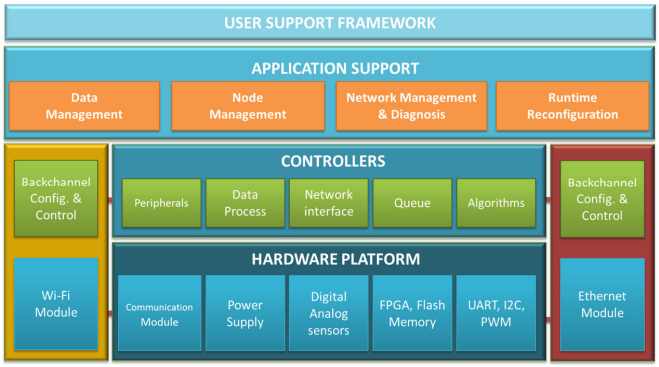

Figure 7. HW-SW support platform.

CEI_Reconfig: This packet includes the capabilities regarding remote reprogramming and system recovery, by receiving the configuration file through serial interface (SPI or UART) from either Wi-Fi or Ethernet. The testbed bootloader is also included in these libraries, which integrates the program memory access depending on the segment to be modified/replace, memory erase, memory reprogramming, verification, and execution in case of needed.

CEI_Console: in combination with *CEI_Reconfig*, it provides remote options for user segment edition and application code execution.

CEI_Linker: This packet contains information regarding the correlation between the library segments and the user dynamic area, as well as initialization directives to support the segmented architecture.

CEI_Diagnosis: this packet implements the functional blocks regarding network diagnosis and runtime deployment performance evaluation.

***CEI_802.15.4*:** it contains the implementation of MAC algorithms and control directives according to the well-known CC2420 module functionality [17], which is also integrated in the Cookie platform, in addition to ZigBee-based modules already implemented.

CEI_AODV: this set of libraries have been developed on top of the *CEI_802.15.4* functionalities in order to provide an AODV-based protocol to deeply experiment with routing capabilities over the wireless network as a use case of the testbed infrastructure. A modified AODV has been implemented due to its flexibility and the capabilities of creating mesh-based topologies.

CEI_Framework: this packet represents the backbone of the memory-segmentation-based architecture and the support of the testbed platform, which is to be used to create new experiments and application tests as well as assure the effectiveness of the partial remote reprogramming. Therefore, it provides users with the Cookie HW-SW support platform for the testbed infrastructure usage.

A general view of the integration of these new support libraries with the HW-SW platform is shown in fig. 7, on which users have the possibility of interacting with the system from different abstraction levels.

V. EXPERIMENTS & TEST CASES

Three main experimental scenarios have been taken into account in order not only to verify and validate the HW-SW implementation of the proposed design, but also put the solution in real application contexts and different conditions so that the architecture usage and performance can be evaluated. Although the server-side interface is not within the scope of the present work, it is important to highlight that the experiments and test cases have been carried out by accessing remotely through a console-based communication or by creating custom GUIs adapted for every particular case. The experimental tests are then classified depending on the type of features to be tested through a real scenario, as follows:

5.1. Scenario1: Backchannel capability-energy performance

In order to test and analyze de performance of the testbed infrastructure, an indoor WSN testbed deployment has been carried out at CEI–UPM, which includes the testbed backchannel interface by using EtherCookie and Wi-Fi-based nodes to remotely access to the platform capabilities, as shown in fig. 8.a). Moreover, the idea goes beyond testing the remote interface, because the memory-segmentation-based architecture has been validated and the energy performance of the nodes was analyzed as well. Two processing layers have been included in the experiments, the first one that includes an ADuC841 microcontroller from Analog Devices [18] and the second one with a C8051F930 from Silicon Labs [19].

In this particular test case, the memory segmentation schema was distributed as shown in fig. 8.b), where 40 KB were reserved to the user dynamic area for experimental purposes, dividing this memory space into 4 KB per slot so that up to ten different tests can be downloaded into the reprogramming architecture (which is fully configurable depending on the user requirements, as explained before). Moreover, from the position 0xB000 to 0xDFFF the support area was also included, which contains the software libraries for providing users which the Cookie HW-SW management capabilities, as well as the recovery segment from 0xE000 for the remote reprogramming and back-up features.

The behavior of the network is as follows: Nodes are configured with the corresponding parameters regarding WSN deployment and the testbed capabilities, and then the basic application starts running (ZigBee-based communication for the wireless connectivity of the nodes). Nodes send information of their sensor measurements every 2 seconds and, after 5 consecutive data transmissions, the nodes are configured to enter in sleep modes. From the point of view of energy modeling and assessment, a power consumption characterization has been carried out by using the on-board circuit included in the Cookie platform for measuring the

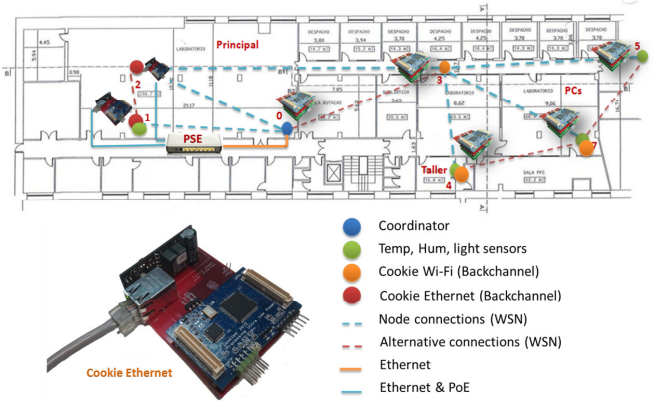

Figure 8.a) Testbed deployment at CEI.

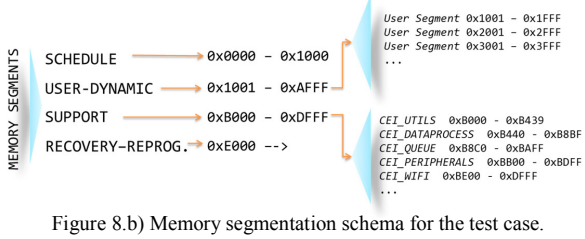

Figure 8.b) Memory segmentation schema for the test case.

consumption of the nodes, as shown in fig. 9, where transitions among the different above-mentioned stages are highlighted. The first area corresponds to the module configuration (the Wi-Fi-based node in the figure), which takes 1.4s and 2.24s during the connection to an access point of the CEI lab. The average consumption of the ADuC841-based nodes is 68 mA, whereas in case of the C8051F930-based nodes is 40mA. As seen in the figure, there is a time transition between configurations that corresponds to the restart and synchronization process of the Wi-Fi module, which is a common value for the whole deployment. Moreover, current picks regarding packet transmission process are highlighted, which mainly correspond to the power consumption of the communication module.

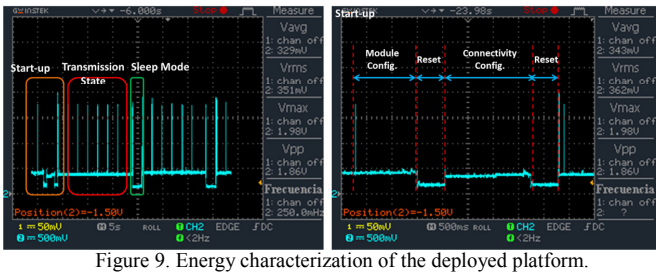

Figure 9. Energy characterization of the deployed platform.

Configurations regarding TCP and UDP connectivity were validated in order to remotely communicate with the nodes by using the backchannel hardware boards. From the point of view of the remote and partial reprogramming by using the proposed architecture, the tests consisted on sending a whole reprogramming file which included the set of software libraries and packets, and compared it with a partial reprogramming which contained only the top level application functionalities, considering that the libraries have been pre-downloaded as an integral part of the system. This scenario allows validating both the custom bootloader/recovery segment as well as the rest of the memory areas by accessing and editing then remotely. In fig. 10, the remote reprogramming process is compared in both whole and partial reprogramming, obtaining a very important optimization in terms of energy and time consumption during the reconfiguration tasks, which therefore represents an essential contribution to the capabilities and effectiveness of the testbed infrastructure, proposed in this work. In table 1, a comparison between the whole and partial reprogramming regarding file size and timing is done for both processing layers.

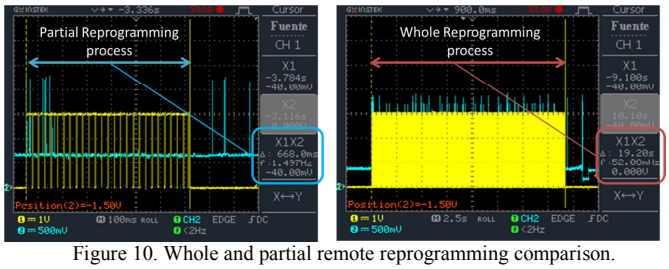

Figure 10. Whole and partial remote reprogramming comparison.

Table 1. performance of the remote reprogramming technique.

Remote Reprogramming / Processing Layer	ADuC841		C8051F930	
	Time	Programming File (kB)	Time	Programming File (kB)
Partial Reprogramming	**664 ms**	**1,14**	**952 ms**	**1,63**
Whole Reprogramming	19,20 s	32,5	21,70 s	36,8

According to the experimental results of the remote reprogramming tests, the time expended to send a single byte remotely is calculated based on the equation 4, taking into account that the main bottleneck in this process is the serial interface between the backchannel communication module and the processing element. For instance, a performed test with the Wi-Fi module for reprogramming the microcontroller through the UART interface configured in 19200 bauds, obtains a value of 570us/byte during the remote reprogramming process, which can be optimized by using higher baud rates or the SPI interface.

$$Pr = \frac{Tp}{Sp} \tag{4}$$

Where Pr is the ratio of reprogramming time spent per byte, Tp is the total amount of time for the partial reprogramming, and Sp is the size of the reprogramming file in bytes.

5.2. Scenario 2: Planning Tool use case–Network analysis

In this experimental test, the main idea is to evaluate and compare simulated deployments of a planning tool with a real indoor environment such as the CEI lab, based on the proposed testbed platform. Three main capabilities of the testbed infrastructure are validated. First, by using the Wi-Fi based Cookie nodes, the mobility over the area of interest is proved to be much flexible without reducing the performance of the testbed (in case of a planning tool it is important to perform an iterative process by changing the position of the nodes in order to compare those deployments with the generated simulations). Second, the network diagnosis techniques have been applied in this particular case in order to obtain real information of the behavior of the nodes in terms of connectivity, quality of the coverage, etc. Third, the remote and partial reprogramming capabilities have also used in order to change application/network set-ups, such as module configurations, power mode strategies, nodes synchronization, among other parameters.

One of the deployments, composed of 18 nodes, is shown in fig. 11 (top), along with the real routing maps generated by applying the proposed network diagnosis, as well as the path assessment – PLR computation for every single node communication (from node source to the coordinator node), as shown in fig. 11 (bottom).

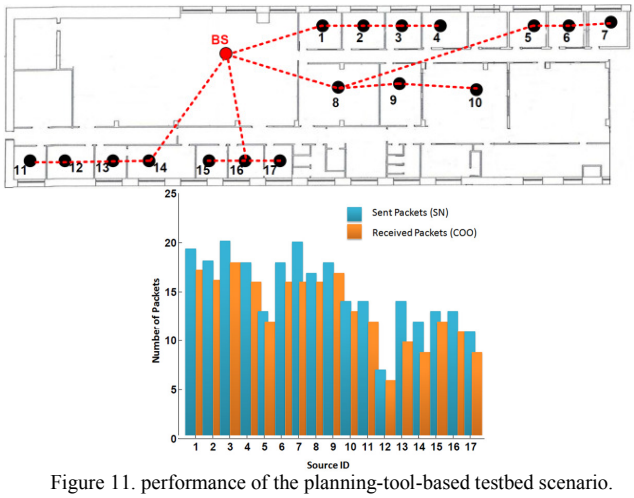

Figure 11. performance of the planning-tool-based testbed scenario.

5.3. Scenario 3: Protocol assessment–debugging capabilities

This third use case is proposed to evaluate the implementation of an AODV-based routing protocol on top of the IEEE 802.15.4 by using the testbed capabilities and specially the debugging tasks and network diagnosis. In this way, four main features have been taken into account for the

evaluation of the implementation. First, by using the interface debugging blocks, the signals related to the connectivity from the microcontroller to the CC2420-based communication module are decoded and obtained, so that specific peaces of low abstraction level information regarding the status of the packet transmission/reception are acquired, as well as verify that the MAC and PHY layers are generating the corresponding frames and the software support component is performing the CSMA/CA algorithm for data transmission in a proper fashion. Second, by using the proposed remote debugging block, specific software components are run in order to analyze the behavior of the node in the corresponding conditions, such as the generation of the routing tables of the protocol, packet processing y differentiating those related to acknowledges, internal protocol messages or application messages, etc. Third, the network diagnosis techniques are applied in this particular case in order to analyze how nodes are generating the different routes in order to transmit packets, as well as study the behavior of the deployment in a crowded-based environment where all the nodes are sending messages randomly to different destinations with a high transmission frequency. Finally, the status of the sensor signals is also monitored in order to verify the behavior of the top level application. As a result, several deployment configurations have been carried out in order to test the software implementation and the generated topologies, as well as measure the PLR.

A mesh-based deployment configuration composed of 9 nodes is shown in fig. 12. Two main configurations have been reprogrammed in order to test the performance of the AODV-based implementation by using the testbed infrastructure. The first one in which node ID 1 sends one message per second (100 messages) to the node ID 9 without any type of traffic (the rest of the nodes only perform routing capabilities) and the second one with an intensive traffic where the nodes send randomly messages every second to the rest of the nodes. Table 2, shows the results obtained from the testbed platform during the experiments, particularly to calculate the PLR (almost 100% of effectiveness without traffic, and more than 99% in traffic conditions) and the specific paths in every transmission process.

VI. Conclusions and Future Work

A complete testbed infrastructure for debugging, testing, managing and analyzing Cookie-based wireless sensor network prototypes and new developments has been designed and implemented, not only aiming to have a robust and reliable laboratory support tool for experiments, but also to increase the efficiency, connectivity and remote control of the wireless nodes in order to provide users an effective way to interact with the platform anywhere at any time, hence joining

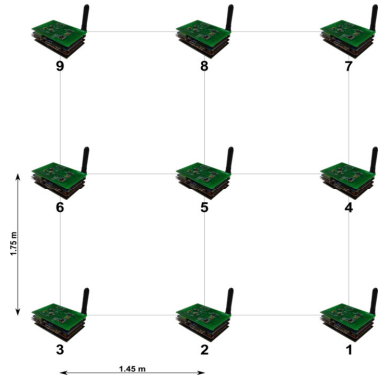

Figure 12. Deployment configuration of the third scenario.

Table 2. Experimental results based on the testbed platform.

1 → 9	Sent packets		Received Packets		Last route	
attempt	Without traffic	With traffic	Without traffic	With traffic	Without traffic	With traffic
1	100	100	100	99	1→5→9	1→5→9
2	100	100	100	100	1→5→9	1→5→9
3	100	100	100	100	1→5→8→9	1→5→9

key points to the future applications in this research field. A heterogeneous network based on high performance connectivity (including Ethernet and Wi-Fi communication) has been proposed and implemented, providing a complete interface for developers to feedback the ZigBee and IEEE 802.15.4 based WSN deployments.

As future approaches for continuing potentiating the usability of the proposed testbed architecture, one of the key features to be implemented is the GUI, so that users can access and control the testbed in a more intuitive way, apart from the web interface in which users can perform experiments remotely outside of the CEI laboratory, through web-based internet connectivity. Moreover, as the HW-SW platform has been created for being scalable, the deployment of more amounts of nodes along the facilities of the CEI lab is possible and feasible, so the next step is to increase the number of nodes connected to the testbed infrastructure.

References

[1] S. Agrawal, M.L. Das, "*Internet of Things — A paradigm shift of future Internet applications*", Nirma University International Conference on Engineering, NUiCONE 2011. pp. 1-7.

[2] S. Krco, M. Nati, D. Pfisterer, N. Mitton, T. Razafindralambo "*A survey on facilities for experimental internet of things research*" in IEEE Communication Magazine, Vol. 49, pp. 58-67. November 2011.

[3] J. Portilla, A. de Castro, E. de la Torre, T. Riesgo, "*A Modular Architecture for Nodes in Wireless Sensor Networks*" in JUCS, vol. 12, nº 3, pp. 328-339, March 2006.

[4] G. Werner-Allen, P. Swieskowski, M. Welsh, "*Motelab: a wireless sensor network testbed*" in Information Processing in Sensor Networks, IPSN 2005, pp. 483 – 488.

[5] E. Ertin, A. Arora, R. Ramnath, V. Naik, S. Bapat, V. Kulathumani, M. Sridharan, H. Zhang, H. Cao, M. Nesterenko, "*Kansei: a testbed for sensing at scale*" in Information processing in sensor networks, IPSN 2006, pp. 399–406.

[6] R. Crepaldi, S. Friso, A. Harris, M. Mastrogiovanni, C. Petrioli, M. Rossi, A. Zanella, and M. Zorzi, "*The design, deployment, and analysis of signetlab: A sensor network testbed and interactive management tool*" in TridentCom 2007. pp. 1 –10.

[7] J. Slipp, C. Ma, N. Polu, J. Nicholson, M. Murillo, S. Hussain, "*Winter: Architecture and applications of a wireless industrial sensor network testbed for radio-harsh environments*" in Communication Networks and Services Research Conference, CNSR 2008. pp. 422 –431.

[8] M. Bal, H. Xue, W. Shen, and H. Ghenniwa, "*A testbed for localization and tracking in wireless sensor networks*" in Systems, Man and Cybernetics, SMC 2009. pp. 3581 –3586.

[9] T. Dimitriou, J. Kolokouris, N. Zarokostas, "*Sensenet: a wireless sensor network testbed*" in ACM MSWiM 2007, pp. 143–150.

[10] J. Portilla, T. Riesgo, A. Abril, A. de Castro, "Rapid prototyping for multi-application sensor networking", 12 November 2007, SPIE Newsroom. DOI: 10.1117/2.1200711.0851.

[11] G. Mujica, V. Rosello, J. Portilla, T. Riesgo, *"Hardware-software integration platform for a WSN testbed based on cookies nodes"*, in 38th Annual Conference on IEEE Industrial Electronics Society, IECON 2012. pp. 6013-6018.

[12] Netgear PoE, h*ttp://www.netgear.com/business/products/switches/*

[13] Silvertel PoE products,*http://www.silvertel.com/poe_products.htm*

[14] Bel Fuse components, *http://www.belfuse.com/*

[15] WIZNet korea, TCP/IP Chip, *http://www.wiznet.co.kr/*

[16] Ultra-low power Wi-Fi RN131C module, *http://www.rovingnetworks.com/products/RN131C.*

[17] CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver Datasheet, Chipcon products from Texas Instruments.

[18] Analog Devices, *http://www.analog.com/en/index.html.*

[19] Silicon Laboratories, www.silabs.com